Chapitre 1

MOTS, LANGAGES et EXPRESSIONS

L'objectif de ce chapitre est de fournir une introduction aux modèles utilisés en informatique pour décrire, représenter et effectuer des calculs sur des séquences finies de symboles.

1. DEFINITIONS

- On appelle **alphabet** ou **vocabulaire** un ensemble fini non vide $\sum$ de symboles (lettres de un ou plusieurs caractères),
  Exemple : *{0,1} : alphabet binaire*

- On appelle **chaîne** ou **mot** toute séquence fini d'éléments de $\sum$,
  - On note ε (epsilon) le **mot vide**,
  - On note $\sum^*$ l'ensemble infini contenant tous les mots possibles sur $\sum$,
  - On note $\sum^+$ l'ensemble des mots non vides que l'on peut former sur $\sum$, c-à-d $\sum^+ = \sum^* - \{\varepsilon\}$
  - On note **|m|** la longueur du mot m, c-à-d le nombre de symboles de $\sum$ composant le mot.
  - On note $\sum^{\mathbf{n}}$ l'ensemble des mots de longueur n.

$$\sum^* = \bigcup_{n=0}^{\infty} \sum^n$$

Exemples :

*1- $\sum$=**{a, b, c}**, aaba, bbbacbb, c, ε, ca sont des mots de $\sum^*$, de longueurs respectives 4, 7, 1, 0 et 2*

*2- $\sum$=**{ad, b, c}**, aba n'est pas un mot de $\sum^*$, ad, cad, bc, adad sont des mots de $\sum^*$ de longueurs respectives 1, 2, 2, et 2*

- On note **.** (point), l'opérateur de **concaténation** de deux mots :
  si $u=u_1 \ldots u_n$ (avec $u_i \in \sum$) et $v=v_1 \ldots v_p$ (avec $v_j \in \sum$) alors $u.v = u_1 \ldots u_n v_1 \ldots v_p$
- u.v est noté aussi uv.

- Propriétés de l'opérateur de concaténation

  - |u.v|=|u|+|v|
  - (u.v).w=u.(v.w) (associativité)
  - ε est l'élément neutre pour la concaténation : u.ε= ε.u=u

- On appelle **langage** sur un alphabet ∑ tout sous-ensemble de ∑*

Exemples : ∑=*{a,b,c}*

1. L_1 l'ensemble des mots de ∑ ayant* ***autant de a que de b.***

→ ***L_1={ε, c, ccc, …,ab, ba,…, abccc, acbcc, accbc,…,aabb, abab, baab, accbccbcccca,…}***

2. L_2 l'ensemble de tous les mots de ∑ ayant* ***exactement 4a.***

→ ***L_2 ={aaaa, aaaac, aaaca, aabaa,…}***

- Opérations sur les langages :
  - **Union** : L1UL2 ={w / w∈L1 ou w ∈L2 }
  - **Intersection** : L1∩L2 ={w / w∈L1 et w ∈L2 }
  - **Concaténation** :L1L2 ={w=w1w2 / w1∈L1 et w2 ∈L2 }
  - **Exponentiation** :L^n ={w=w1…wn / wi∈L ∀ i∈{1,…,n} }

    - **$L^n = L^{n-1}L$ (L est concaténé n-1 fois avec lui même)**
    - **$L^0 = \{\varepsilon\}$**
    - **$L^1 = L^0L = L$**
    - **$L^2 = L^1L = LL$**
    - **$L^3 = L^2L = LLL$**
    - **…**

- **Fermeture de Kleene (Étoile)** : $L^* = U_{n\geq 0} L^n$ (un nombre qcq., éventuellement nul, de concaténations de L)
- **Fermeture positive** : $L^+ = U_{n>0} L^n$ (un nombre non nul de concaténations de L)

Exemples : *Soient* ***L={A,B,…,Z, a, b,…,z}*** *et* ***C={0,1,2,…,9}***

1. ***LUC*** *est l'ensemble des lettres et chiffres :* ***LUC={A,B,…,Z,a,b,…,z,0,1,…,9}***
2. ***LC*** *est l'ensemble de mots formés d'une lettre suivi d'un chiffre :* ***LC={A0, A1,…, B8, B9,…,z0,z1,…z9}***
3. L^4 *est l'ensemble de mots de 4 lettres:* L^4***={ABCD,abcd, aaac, bcaa,…}***
4. ***L**** *est l'ensemble de tout les mots de lettres y compris* ε*:* ***L*={ε, ab, abc, il, elle, …}***
5. C^+ *est l'ensemble de tout les mots d'au moins un chiffre :* C^+***={0,12, 950, 25,…}***
6. ***L(LUC)**** *est l'ensemble de tout les mots de lettres et de chiffres commençant par une lettre :* ***L(LUC)* ={a, b3, cde, kj1,… }***

- ∃ plusieurs types de langages pour décrire tous les mots acceptables. Par exemple, en analyse lexicale, on s'intéresse aux langages réguliers.
- Un **Langage Régulier** (**LR**) sur un alphabet ∑ est défini récursivement comme suit :
  - {ε} est un LR sur ∑
  - si a ∈∑, {a} est un LR sur ∑
  - si R est un LR sur ∑, alors R^n et R* sont des LRs sur ∑
  - si R1 et R2 sont des LRs sur ∑ alors R1UR2 et R1R2 sont des LRs

2. EXPRESSIONS REGULIERES / RATIONNELLES

2.1 Expressions régulières basiques

- Les **expressions régulières** (**ER**) sur un alphabet ∑ et les langages qu'elles décrivent sont définis récursivement comme suit :
  - ε est une ER qui décrit le langage {ε}
  - si a∈∑, alors a est une ER qui décrit {a}
  - si r est une ER qui décrit le langage R alors (r) est une ER décrivant le même langage
  - si r est une ER qui décrit le langage R alors (r*) est une ER décrivant R*
  - si r est une ER qui décrit R alors (r+) est une ER décrivant R+
  - si r et s sont des ERs qui décrivent respectivement R et S, alors
    - (r)|(s) est une ER décrivant RUS
    - (r)(s) est une ER décrivant RS
    - Il n'y a pas d'autres ERs

- Priorités décroissantes d'opérateurs (pour éviter des parenthèses en superflus) : *, concaténation, |

Exemple : ***((a)((b)*))|(c) ⇔ ab*|c***

- Des propriétés algébriques peuvent être utilisées pour manipuler les ERs sous des formes équivalentes :
  - r|s=s|r (| est commutatif)
  - (r|s)|t=r|(s|t) (| est associatif)
  - r(s|t)=rs|rt (concaténation est distributive par rapport à |)
  - (r|s)t=rt|st (| est distributif par rapport concaténation)

Exemples : ∑=*{a,b,c}*

1. ***a**** *dénote l'ensemble des mots formés d'un nombre qcq. (éventuellement nul) de a :{ε, **a, aa, aaa,…}***

2. ***a|b*** *dénote l' ensemble des mots formés de a et de b :* ***{a,b}***

3. ***(a|b)*=(b|a)**** *dénote l'ensemble de tous les mots formés d'un nombre qcq. (éventuellement nul) de a ou b :{ε, **a, b , aa, bb, ab, aaab,…}***

4. ***(a|b)(a|b)*** *dénote l'ensemble de tous les mots de a et de b de longueur 2 :* ***{aa,ab, ba, bb}***

5. **((a)((b)*))|(c) = ab*|c** *dénote l'ensemble des mots formés soit d'un seul a suivi de 0 ou plusieurs b soit d'un c :* ***{a, ab, abb, c,… }***

6. **(a)|((b)*(c) = a|b*c** *dénote l'ensemble des mots formés soit d'un a soit de 0 ou plusieurs b suivis d'un c :* ***{a, c, bc, bbc,…}***

2.2 Définitions / Descriptions Régulières

- *Une ER décrivant les identificateurs en C (toute suite non vide de caractères composée de chiffres, de lettres ou du symbole _ et qui ne commencent pas par un chiffre) pourrait être :*

  Identificateur_C=(a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y| Z|_)(a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|0|1|2|3|4|5|6|7|8|9|_)*

- *C'est un peu illisible alors on s'autorise des* ***définitions régulières*** *et le symbole '-' sur des types ordonnés (lettres, chiffres,…).*

- Une **définition régulière (DR)** est une suite de définitions de la forme :

$$\begin{cases} d1 = r1 \\ d2 = r2 \\ \dots \\ dn = rn \end{cases}$$

Où chaque ri est une ER sur l'alphabet ∑U{d1,d2,…,di-1}

- *L'UL **Identificateur_C** peut devenir :*

  Lettre = ***A-Z|a-z***

  Chiffre = ***0-9***

  Sep = _

  Identificateur_C = (Lettre|sep)(Lettre|sep|chiffre)*

Exemple :

- *Les nombres sans signe en Pascal (Ex. : 5280, 39.37, 6.336E4 ou 1.894E-4, ...) peuvent être définis comme suit :*

chiffre	***0-9***
chiffres	***chiffre chiffre****
fraction_op	***. chiffres***|ε
exposant_op	***(E(***+|-|ε***) chiffres)***| ε
nombre	***chiffres fraction_op exposant_op***

- *Une **fraction_op** est soit un **point décimal** suivi d'au moins un **chiffre**, soit manquante.*

- *Un **exposant_op** s'il n'est pas manquant, est un **E** suivi d'un + ou – optionnel, suivi d'au moins un **chiffre**.*

- *Noter qu'au moins un chiffre doit suivre le point décimal, de sorte que nombre ne reconnaît pas 1. mais reconnaît 1.0*

2.3 Notations Abrégées

+: Au moins une fois /instance

a+ est une abréviation de aa*

le + a la même priorité et la même associativité que *

? **: zéro ou une fois**

a? est une abréviation de a|ε

Classes de caractères

[a-z] = a|b|c|…|z

[abc] = a|b|c

d'où la spécification suivante :

chiffre	*0-9*
chiffres	$chiffre^+$
fraction_op	*(.chiffres)?*
exposant_op	*(E(+|-)? chiffres)?*
nombre	*chiffres fraction_op exposant_op*

2.4 Extensions notationnelles

Une ER se compose de caractères normaux et de méta-caractères qui ont une signification spéciale : **$, \, ^, [,], {,}, (,), +, -, *, /, |,?**

C	Tout caractère c qui n'est pas un méta- caractère	a
\c	Si ***c*** est un méta-caractère alors le caractère c littéralement	\+, \.
« s »	La chaîne de caractère ***s*** littéralement	« abc *+ »
r1|r2	***r1*** suivie de ***r2***	Ab
.	N'importe quel caractère, excepté le cacractère « à la ligne »	a.b
^	Comme premier caractère de l'expression, signifie le début de ligne	^abc
$	Comme dernier caractère de l'expression, signifie la fin de ligne	abc$
[s]	N'importe lequel des caractères constituant la chaîne ***s***	[abc]
[^s]	N'importe lequel des caractères	[^abc]
r*	0 ou plusieurs occurrences de ***r***	a*
r+	1 ou plusieurs occurrences de ***r***	a+
r?	0 ou 1 occurrence de ***r***	a?
r{m}	***m*** occurrences de ***r***	a{3}
r{m,n}	***m*** à ***n*** occurrences de ***r***	a{5,8}
r1|r2	***r1*** ou ***r2***	a|b
r1/r2	***r1*** si elle est suivie de ***r2***	ab/cd
(r)	***r***	(a|b)?c
\n	Caractère « à la ligne »	
\t	Tabulation	
{}	Pour faire référence à une DR	{NOMBRE}
EOF	Fin de fichier (uniquement avec flex)	

ATTENTION

- Certains outils font la différence entre les minuscules et majuscules
- **$**, ^ et / ne peuvent apparaître dans des () ni dans des DRs
- ^ perd sa signification *début de ligne* s'il n'est pas au début de l'ER

- **$** perd sa signification *fin de ligne* s'il n'est pas à la fin de l'ER
- à l'intérieur des **[ ]**, seul \ reste un méta-caractère, le **–** ne le reste que s'il n'est ni au début ni à la fin de l'ER.

PRIORITÉS

- a|b* est interprété comme (a)|(b*)
- abc{1,3} est interprété comme (abc){1,3}, dans d'autre outils comme ab(c{1,3})
- ^a|b est interprété comme (^a)|b ou comme ^(a|b)

5. CONCLUSION

Une expression régulière sera compilée en un automate à états finis qui servira de « reconnaisseur » de mots, en particulier, de lexèmes d'un langage de programmation donné.